REMARKS

The specification has been amended for clarification purposes only. Claims 1-4, 7, 20, and 21 have been amended; claim 22 has been added. Therefore, claims 1-9 and 20-22 are currently pending in the case. Further examination and reconsideration of the presently claimed application are hereby respectfully requested.

Objection to the Claims

Claim 1 was objected to for an informality. Statements in the Office Action suggest that the term "uimanager" should be changed to "UIManager" (Office Action, page 2). Claim 1 has been amended to expedite prosecution and to clarify the claim language in a manner that addresses the concerns expressed in the Office Action. Accordingly, Applicants respectfully request removal of this objection.

Section 102 Rejections

Claims 1-9, 20, and 21 were rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,727,918 to Nason (hereinafter "Nason"). The standard for "anticipation" is one of fairly strict identity. A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference. *Verdegaal Bros. v. Union Oil Co. Of California*, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987), MPEP 2131. Nason does not disclose all limitations of the currently pending claims, some distinctive limitations of which are set forth in more detail below.

Nason fails to disclose an application program interface comprising a UIManager containing code which, when executed by a processor, generates either a default image via a first pointer or a user-defined image via a second pointer, and a peer component coupled to the UIManager and configured to: (i) enable a selection capability of the UIManager, wherein said enablement causes the UIManager to select either the first pointer or the second pointer, or (ii) disable the selection capability of the UIManager, wherein said disablement causes the UIManager to select the first pointer. Amended independent claim 1 recites in part:

> A display system, comprising ... an application program interface coupled to the display, wherein the interface comprises: a UIManager containing code which, when executed by the processor, generates either a default image or an image that is user-defined; a first pointer, which points to the code for generating the default image with an operating

> system independent look and feel setting; a second pointer, which points to the code for generating the user-defined image with an operating system dependent look and feel setting; and a peer component coupled to the UIManager and configured to (i) enable a selection capability of the UIManager, wherein said enablement causes the UIManager to select either the first pointer or the second pointer, or (ii) disable the selection capability of the UIManager, wherein said disablement causes the UIManager to select the first pointer.

Support for the amendment to claim 1 may be found in the specification, for example, on page 35, line 27 through page 37, line 14 and Fig. 16.

The specification provides a unique system of software components that enables an application program (e.g., a Java application) to be truly portable across all operating system (OS) platforms by providing various means for maintaining the "look and feel" of the application program. "A software program is said to be 'portable' across various platforms if the program can run without modification on any of those platforms." (Specification, page 6, lines 23-24). Before disclosing such means, the Specification highlights several drawbacks of prior art attempts at application portability.

For example, Java application programs utilize a platform-dependent application program interface (API), commonly known as the Abstract Windowing Toolkit (AWT), to produce heavyweight software components that are written in native code (i.e., instructions specific to a particular OS). When AWT heavyweight software components are used for displaying images, the "look and feel" of those images may differ depending on the particular OS running the Java application program. In the context of a graphical user interface (GUI), "the 'look and feel' of a GUI refers to such things as the appearance, color and behavior of Buttons, TextFields, Listboxes, menus, etc ..." (Specification, page 7, line 4-8). Thus, there are certain limitations within the AWT that prohibit true portability of Java application programs, especially in terms of the look and feel of the application program. *See*, e.g., Specification, page 7, line 24 to page 9, line 10, and page 18, line 10 to page 19, line 23.

Swing was developed in an effort to overcome the platform-dependency of Java application programs. An API written using Swing contains no native code, and therefore, can be run on substantially any OS without changing the look and feel of the application. *See*, e.g., Specification, page 9, lines 14-30, and page 19, line 25 to page 20, line 4. Unfortunately, the lightweight software components of Swing cannot completely eliminate the platform-dependency of Java applications that use AWT. Since Swing versions of many AWT components (e.g., containers, such as frames) are unavailable, many programmers have attempted to mix Swing and AWT components within a given API. *See*, e.g., Specification, page 21,

lines 1-13. However, straightforward mixing of Swing and AWT components tends to create many problems that programmers have simply learned to accept. *See*, e.g., Specification, page 22, lines 1-28.

For example, before the present invention was introduced, it was simply impossible to maintain the original "look and feel" of a legacy application containing AWT-based controls (or a mixture of AWT and Swing-based controls), when Swing counterparts were used to replace the AWT-based controls. In conventional solutions, the original "look and feel" of AWT-based controls were simply replaced with the global "look and feel" settings of their Swing counterparts. *See*, e.g., Specification, page 35, line 27 to page 36, line 8.

In order to preserve the original "look and feel", or provide more than one distinct "look and feel" within a give application program, the presently claimed case provides a system of software components – referred to as "AWTSwing" components – that enable lightweight Swing-based controls to be used in place of the heavyweight AWT-based controls initially used for displaying objects. More specifically, aspects of the presently claimed case may enable a Swing-based control to be used for displaying an object, which was initially created by an AWT-based control, while preserving the original AWT "look and feel" of the object. This is in direct contrast to the conventional Swing API, whose global "look and feel" settings are applied to all Swing objects. *See*, e.g., Specification, page 36, line 10 to page 37, line 14 and FIG. 16.

In one embodiment of the present invention, the "look and feel" of a displayed object may depend on the particular thread (e.g., Swing or AWT) originally used to create the object. For example, if an object is initially created with a Swing-based control, the object may be displayed with a default image, whose "look and feel" is associated with the Swing-based control, and therefore, independent from the operating system. On the other hand, if an object is initially created with an AWT-based control, but later replaced by a Swing-based control, the object may be displayed with a user-defined image, whose "look and feel" is associated with the original AWT-based control, and therefore, dependent on the operating system.

With regard to present claim 1, the presently claimed application program interface (API) includes a UIManager (278, FIG. 16) containing code which, when executed by the processor, generates either a default image or an image that is user-defined. More specifically, the UIManager may be provided with a capability (Set Thread Look and Feel method 280) for selecting either a first pointer or a second pointer.

As shown in FIG. 16 and described in present claim 1, the first pointer may point to code for generating the default image with an operating system independent look and feel setting (i.e., Default Look and Feel 282), whereas the second pointer may point to code for generating the user-defined image with an operating system dependent look and feel setting (i.e., Custom Look and Feel 284). The presently claimed application program interface also includes a peer component (AWTSwing Peer 274), which is coupled to the UIManager for: (i) enabling the selection capability of the UIManager, wherein said enablement causes the UIManager to select either the first pointer or the second pointer, or (ii) disabling the selection capability of the UIManager, wherein said disablement causes the UIManager to select the first pointer.

In contrast to the presently claimed case, Nason fails to disclose an application program interface including: a UIManager for generating either a default image via a first pointer or a user-defined image via a second pointer, and a peer component coupled to the UIManager for: (i) enabling a selection capability of the UIManager, wherein said enablement causes the UIManager to select either the first pointer or the second pointer, or (ii) disabling the selection capability of the UIManager, wherein said disablement causes the UIManager to select the first pointer.

Statements in the Office Action suggest that Nason provides teaching for a "UIManager containing code which, when executed by the processor generates either a default image or an image [that] is user defined... in column 5, lines 45-63" (Office Action, page 3). The Applicant respectfully disagrees for at least the reasons set forth in more detail below.

In column 5, lines 45-63, Nason describes an alternate display content controller (6, Fig. 1) that interacts with a computer operating system (5B) and hardware drivers (5C) to control allocation of display space (1) and create one or more parallel graphical user interfaces (e.g., 2, 2A, 2B, 4), which may be displayed adjacent to the operating system desktop (3). Nason discloses that "[a]s software, [the] alternative display content controller may be an application program running on the computer operating system, or may include an operating system kernel of varying complexity ranging from dependent on the native operating system ... to a parallel system independent of the native operating system" (Nason, column 5, lines 52-59). As such, Nason describes one embodiment of the alternate display content controller, in which the controller is implemented in software, which may itself be independent from, or dependent on, the native operating system.

However, a software application (i.e., the alternate display content controller of Nason) -- which is independent from or dependent on a native operating system -- is NOT equivalent to a software application (i.e., the presently claimed UIManager), which is capable of generating images with operating system independent and dependent "look and feel" settings. In other words, the alternate display content controller described by Nason cannot be used to provide teaching for the presently claimed UIManager, which as noted above, contains code for generating either a default image (with an operating system independent look and feel setting) or a user-defined image (with an operating system dependent look and feel setting). The alternate display content controller of Nason is not disclosed as capable of generating images with operating system independent and dependent "look and feel" settings, and therefore, cannot be used to anticipate the presently claimed UIManager.

In addition to the presently claimed UIManager, Nason fails to provide teaching for the presently claimed peer component, which is coupled to the UIManager for: (i) enabling a selection capability of the UIManager, wherein said enablement causes the UIManager to select either the first pointer or the second pointer, or (ii) disabling the selection capability of the UIManager, wherein said disablement causes the UIManager to select the first pointer.

As noted above, statements in the Office Action suggest that the alternate display content controller of Nason is somehow equivalent to the presently claimed UIManager. The Applicant disagrees, for at least the reasons set forth above. However, for the sake of argument, the Examiner's assertion will be considered in the arguments presented below to show the lack of teaching within Nason for the presently claimed peer component and it's association with the presently claimed UIManager.

Even if one were to falsely equate the alternate display content controller of Nason to the presently claimed UIManager, there is simply no teaching within Nason for the presently claimed peer component, which is coupled to the so-called "UIManager" of Nason for: (i) enabling a selection capability of the UIManager, wherein said enablement causes the UIManager to select either a first pointer or a second pointer, or (ii) disabling the selection capability of the UIManager, wherein said disablement causes the UIManager to select the first pointer. For example, the alternate display content controller of Nason (i.e., the alleged UIManager) does not include a "selection capability," much less one that enables the alternate display content controller to select either a first pointer (for generating a default image with an operating system independent look and feel setting) or a second pointer (for generating a user-defined image with an operating system dependent look and feel setting). If no selection capability exists, then Nason certainly

cannot provide a peer component for enabling/disabling a non-existent selection capability of a non-existent UIManager. For at least these reasons, the teachings of Nason cannot be relied upon to anticipate the presently claimed peer component.

Nason fails to disclose a computer-readable storage device including an application program comprising a mixture of Swing and AWT-based controls for generating graphical representations of objects. Amended independent claim 20 recites, in part, "[a] computer-readable storage device, comprising ... an application program running under the operating system and comprising a mixture of Swing and AWT-based controls for generating graphical representations of objects ..." Support for the amendments made to claim 21 may be found in Fig. 16 of the drawings and supporting text.

Nason simply fails to disclose an application program comprising a mixture of Swing and AWT-based controls for generating graphical representations of objects. Though Nason discloses that the "alternative display content controller may also include content and operating software such as JAVA delivered over the Internet" (Nason, column 5, lines 61-63), such disclosure provides absolutely no indication that the alternative display content controller of Nason may include a mixture of Swing and AWT-based controls. Therefore, the teachings of Nason cannot be used to anticipate the presently claimed application program.

In fact, there is simply no mention of Swing or AWT-based controls within the teachings of Nason. Nor is there any teaching, suggestion or desirability within Nason for mixing, substituting or converting existing AWT-based controls to Swing-based controls (or vice versa). Therefore, in anticipation of the Examiner's next rejection, Applicants assert that the teachings of Nason cannot be combined or modified with other references, which may disclose techniques for mixing Swing and AWT components, since Nason provides absolutely no motivation or desirability for such combination or modification.

Nason fails to disclose a system of software components invoked during runtime and adapted for: displaying a graphical representation of a first object, which is created by the application program with a first characteristic appearance and behavior, if the graphical representation of the first object is generated by a Swing-based control, and displaying a graphical representation of a second object, which is created by the application program with a second characteristic appearance and behavior, distinct from the first, if the graphical representation of the

second object is generated by an AWT-based control. Amended independent claim 20 also recites in part:

> A computer-readable storage device, comprising... a system of software components invoked during runtime and adapted for: displaying a graphical representation of a first object, which is created by the application program with a first characteristic appearance and behavior, if the graphical representation of the first object is generated by a Swing-based control; and displaying a graphical representation of a second object, which is created by the application program with a second characteristic appearance and behavior, distinct from the first, if the graphical representation of the second object is generated by an AWT-based control.

Support for the amendments made to claim 21 may be found in Fig. 16 of the drawings and supporting text.

As noted above, Nason fails to mention the existence or use of Swing and/or AWT-based controls. In addition, Nason fails to disclose that images (or graphical representations of objects) may be generated with operating system independent or dependent "look and feel" settings. For at least these reasons, Nason cannot be relied upon to provide teaching for a system of software components, which are configured for displaying a graphical representation of a first object with a first characteristic appearance and behavior (i.e., a first "look and feel"), if the graphical representation of the first object is generated with a Swing-based control, and displaying a graphical representation of a second object with a second characteristic appearance and behavior (i.e., a second "look and feel"), distinct from the first, if the graphical representation of the second object is generated with an AWT-based control.

For at least the reasons set forth above, Nason fails to anticipate all limitations of independent claims 1 and 20. Therefore, claims 1 and 20, as well as claims dependent therefrom, are asserted to be patentably distinct over Nason. Accordingly, removal of this rejection is respectfully requested.

Patentability of the Added Claim

The present amendment adds claim 22 which is dependent from claim 1, and is patentably distinct over the cited art for at least the same reasons as claim 1. Accordingly, allowance of added claim 22 is respectfully requested.

CONCLUSION

This response constitutes a complete response to all issues raised in the Office Action mailed October 5, 2004. In view of the remarks traversing rejections, Applicants assert that pending claims 1-9, and 20-22 are in condition for allowance. If the Examiner has any questions, comments, or suggestions, the undersigned earnestly requests a telephone conference.

No fees are required for filing this amendment; however, the Commissioner is authorized to charge any additional fees which may be required, or credit any overpayment, to Daffer McDaniel, LLP Deposit Account No. 50-3268/5468-08300.

Respectfully submitted,

Mollie E. Lettang

Mollie E. Lettang
Reg. No. 48,405
Agent for Applicant(s)

Daffer McDaniel, LLP
P.O. Box 684908
Austin, TX 78768-4908
Ph: (512) 476-1400
Date: January 3, 2005
JMF